Bonnesoeur Maxime
Gautier Maxime
Furrer Stanislas

Student names: Bonnesoeur Maxime, Gautier Maxime, Furrer Stanislas

Instructions: Update this file (or recreate a similar one, e.g. in Word) to prepare your answers to the questions. Feel free to add text, equations and figures as needed. Hand-written notes, e.g. for the development of equations, can also be included e.g. as pictures (from your cell phone or from a scanner). ***This lab is graded.*** *and must be submitted before the* ***Deadline : 11-04-2018 Midnight****.*
Please submit both the source file (.doc/*.tex) and a pdf of your document, as well as all the used and updated Python functions in a single zipped file called lab6_name1_name2_name3.zip where name# are the team member's last names. Please submit only one report per team!*

The file `lab#.py` is provided to run all exercises in Python. The list of exercises and their dependencies are shown in Figure 1. When a file is run, message logs will be printed to indicate information such as what is currently being run and and what is left to be implemented. All warning messages are only present to guide you in the implementation, and can be deleted whenever the corresponding code has been implemented correctly.

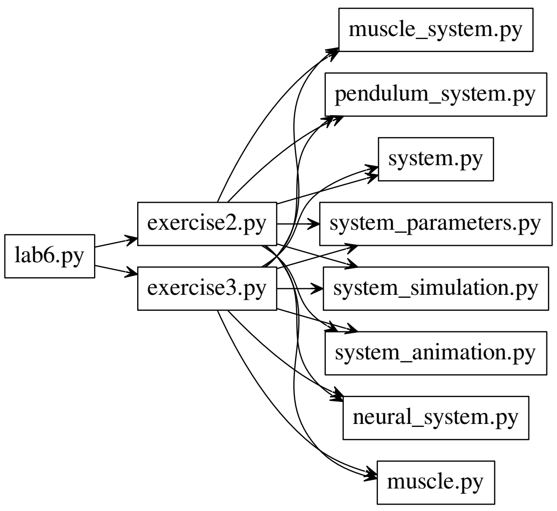


Figure 1: Exercise files dependencies. In this lab, you will be modifying `exercise1.py` and `pendulum_system.py`

Files to complete the exercises

- `lab6.py` : Main file
- `exercise2.py` : Main file to complete exercise 2
- `exercise3.py` : Main file to complete exercise 3
- `system_parameters.py` : Parameter class for Pendulum, Muscles and Neural Network (Create an instance and change properties using the instance. You do not have to modify the file)
- `muscle.py` : Muscle class (You do not have to modify the file)
- `system.py` : System class to combine different models like Pendulum, Muscles, Neural Network (You do not have to modify the file)

Bonnesoeur Maxime
Gautier Maxime
Furrer Stanislas

- `pendulum_system.py` : Contains the description of pendulum equation and Pendulum class. You can use the file to define perturbations in the pendulum.
- `muscle_system.py` : Class to combine two muscles (You do not have to modify the file)
- `neural_system.py` : Class to describe the neural network (You do not have to modify the file)
- `system_simulation.py` : Class to initialize all the systems, validate and to perform integration (You do not have to modify the file)
- `system_animation.py` : Class to produce animation of the systems after integration (You do not have to modify the file)

NOTE : '*You do not have to modify*' does not mean you should not, it means it is not necessary to complete the exercises. But, you are expected to look into each of these files and understand how everything works. You are free to explore and change any file if you feel so.

Exercise 2 : Pendulum model with Muscles

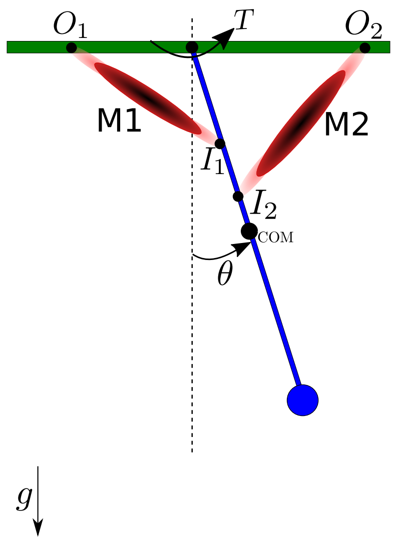


Figure 2: Pendulum with Antagonist Hill Muscles

The system is comprised of a physical pendulum described by equation 1 and a pair of antagonist muscles **M1** and **M2**. Muscle **M1** extends the pendulum (θ increases) and Muscle **M2** flexes the muscle (θ decreases).

Consider the system only for the pendulum range $\theta = [-\pi/2, \pi/2]$

$$I\ddot{\theta} = -0.5 \cdot m \cdot g \cdot L \cdot sin(\theta) \tag{1}$$

Where,

- I - Pendulum inertia about the pendulum pivot joint $[kg \cdot m^2]$
- θ - Pendulum angular position with the vertical $[rad]$
- $\ddot{\theta}$ - Pendulum angular acceleration $[rad \cdot s^{-2}]$
- m - Pendulum mass $[kg]$
- g - System gravity $[m \cdot s^{-2}]$

- L - Length of the pendulum $[m]$

Each muscle is modelled using the Hill-type equations that you are now familiar with. Muscles have two attachment points, one at the origin and the other at the insertion point. The origin points are denoted by $O_{1,2}$ and the insertion points by $I_{1,2}$. The two points of attachment dictate how the length of the muscle changes with respect to the change in position of the pendulum.

The active and passive forces produced by the muscle are transmitted to the pendulum via the tendons. In order to apply this force on to the pendulum, we need to compute the moment based on the attachments of the muscle.

Using the laws of sines and cosines, we can derive the length of muscle and moment arm as below. The reference to the paper can be found here Reference,

$$L_1 = \sqrt[2]{a_1^2 + a_2^2 + 2 \cdot a_1 \cdot a_2 \cdot \sin(\theta)} \tag{2}$$

$$h_1 = \frac{a_1 \cdot a_2 \cdot \cos(\theta)}{L_1} \tag{3}$$

Where,

- L_1 : Length of muscle 1
- a_1 : Distance between muscle 1 origin and pendulum origin ($|O_1C|$)
- a_2 : Distance between muscle 1 insertion and pendulum origin ($|I_1C|$)
- h_1 : Moment arm of the muscle

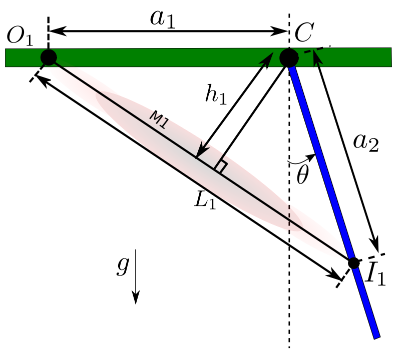


Figure 3: Computation of muscle length and moment arm

Equation 2 can be extended to the Muscle 2 in similar way. Thus, the final torque applied by the muscle on to the pendulum is given by,

$$\tau = F \cdot h \tag{4}$$

Where,

- τ : Torque $[N \cdot m]$
- F : Muscle Tendon Force $[N]$
- h : Muscle Moment Arm $[m]$

In this exercise, the following states of the system are integrated over time,

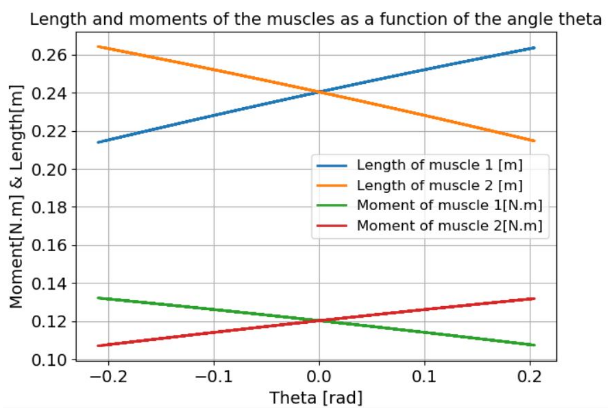
$$X = \begin{bmatrix} \theta & \dot{\theta} & A_1 & l_{CE1} & A_2 & l_{CE2} \end{bmatrix} \tag{5}$$

Where,

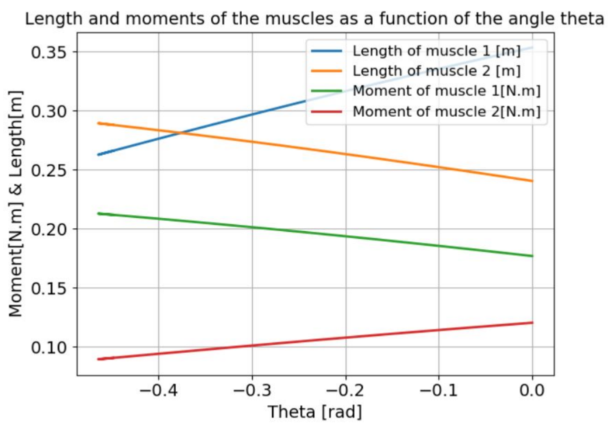
- θ : Angular position of the pendulum [rad]
- $\dot{\theta}$: Angular velocity of the pendulum [rad/s]
- A_1 : Activation of muscle 1 with a range between [0, 1]. 0 corresponds to no stimulation and 1 corresponds to maximal stimulation.
- l_{CE1} : Length of contracticle element of muscle 1
- A_2 : Activation of muscle 2 with a range between [0, 1]. 0 corresponds to no stimulation and 1 corresponds to maximal stimulation.
- l_{CE2} : Length of contracticle element of muscle 2

To complete this exercise you will make use of the following files, `exercise2.py`, `system_parameters.py`, `muscle.py`, `system.py`, `pendulum_system.py`, `muscle_system.py`, `system_simulation.py`

2a. For a given set of attachment points, compute and plot the muscle length and moment arm as a function of θ between $[-\pi/4, \pi/4]$ using equations in eqn:2 and discuss how it influences the pendulum resting position and the torques muscles can apply at different joint angles. You are free to implement this code by yourself as it does not have any other dependencies.



(a) Symmetric Muscles



(b) Asymmetric Muscles

Figure 4: Muscle Length and Moment as a function of θ

We computed the muscle length and moment as a function of θ for different attachment points. Figure 8a is for attachment points where both a_1 and a_2 from Figure 3 are the same length : 17 cm. Figure 8b is for attachment points where a_1 and a_2 of the left muscle are 25cm and Muscle 2 is kept the same. We can see in both graph an equilibrium position in the point where both length are identical and their curves cross. The equilibrium for a symmetrical system is evidently when $\theta = 0rad$ and for the asymmetrical system when $\theta = -0.37rad$. The equilibrium, or resting position is when both muscles have the same length because since they have the same properties, they will exert the same force in opposite directions and the pendulum will not move unless the muscles are stimulated.

The position of the muscle's attachment points affect the moment of the muscles and the torque in a similar manner. The torque will drive the pendulum to the resting position but constant stimulation will keep it from stopping there. Since the position of the muscles determine the resting position, they also play a role with the torque.

If muscles are too long, or in other words if the attachment points are too far away from the pendulum's attachment point C, the muscles are not able to exert a contracting force and the pendulum remains almost immobile, as can be seen in Figure 5. This is due to the fact that when the contractile element is over stretched, only the passive force is present and this force is not activated by stimulation but constant, so the pendulum will not move. The slight oscillation is due to the very small active force since the muscle is stretched to the limit at which active force can be generated : a_1 and a_2 is 20cm for both muscles.

Proper positioning of the muscle's attachment points is essential to create a viable and controllable system since it determines intrinsic properties like resting position or pendulum torque.

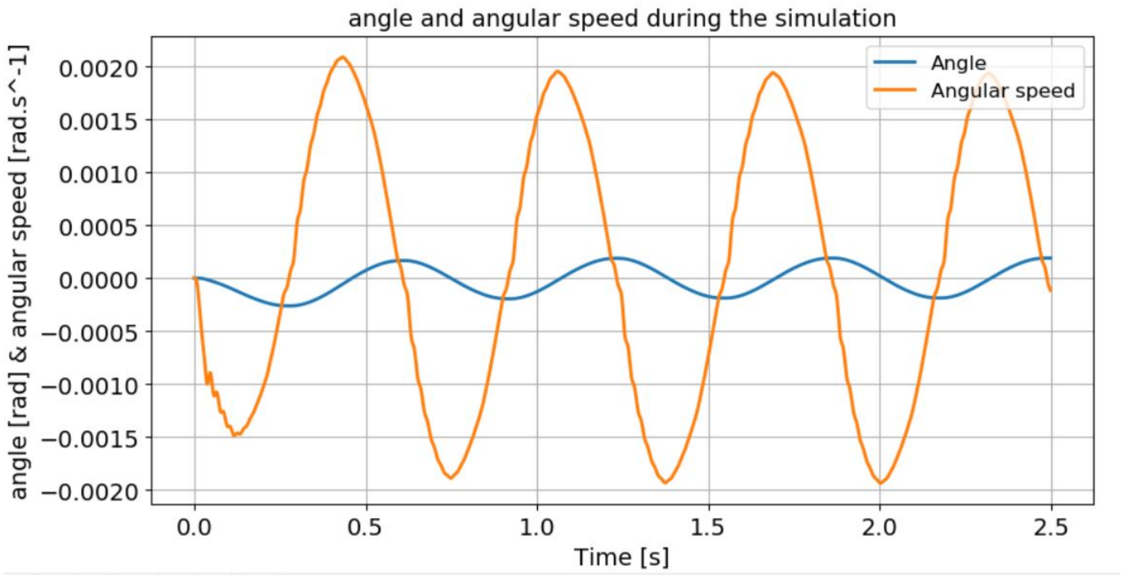
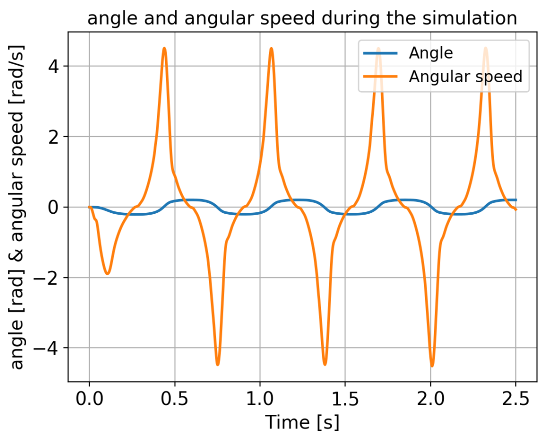
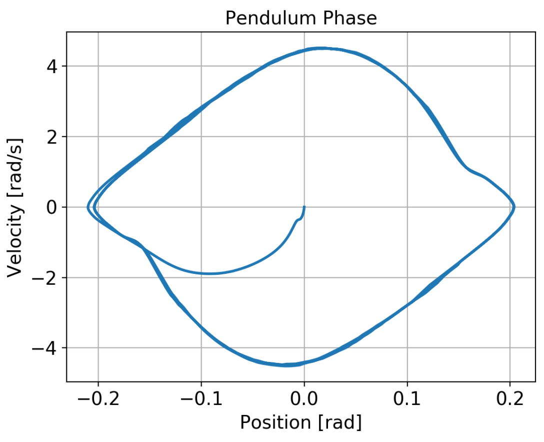


Figure 5: Pendulum with overstretched muscles

2b. Using simple activation wave forms (example : sine or square waves) applied to muscles (use `system_simulation.py::add_muscle_activations` method in `exercise2.py`), try to obtain a limit cycle behavior for the pendulum. Use relevant plots to prove the limit cycle behavior. Explain and show the activations wave forms you used. Use `pendulum_system.py::PendulumSystem::pendulum_system` function to perturb the model.



(a) State of Pendulum



(b) Phase of Pendulum

Figure 6: Proof of limit cycle behavior

Bonnesoeur Maxime
Gautier Maxime
Furrer Stanislas

We can see in Figure 6 the State and phase of the pendulum when there is no perturbation. The activation functions we used are simple identical sinusoidal functions with opposite frequencies. We can see in Figure 7 the activation functions we used as well as the muscle response.

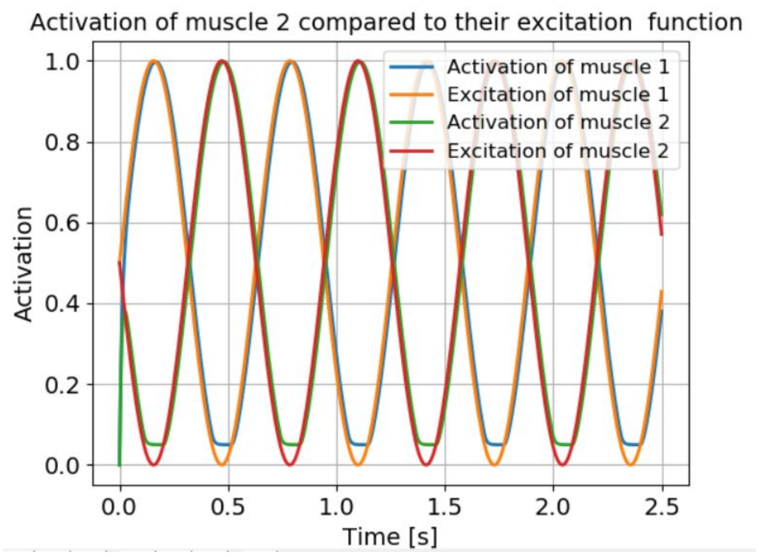
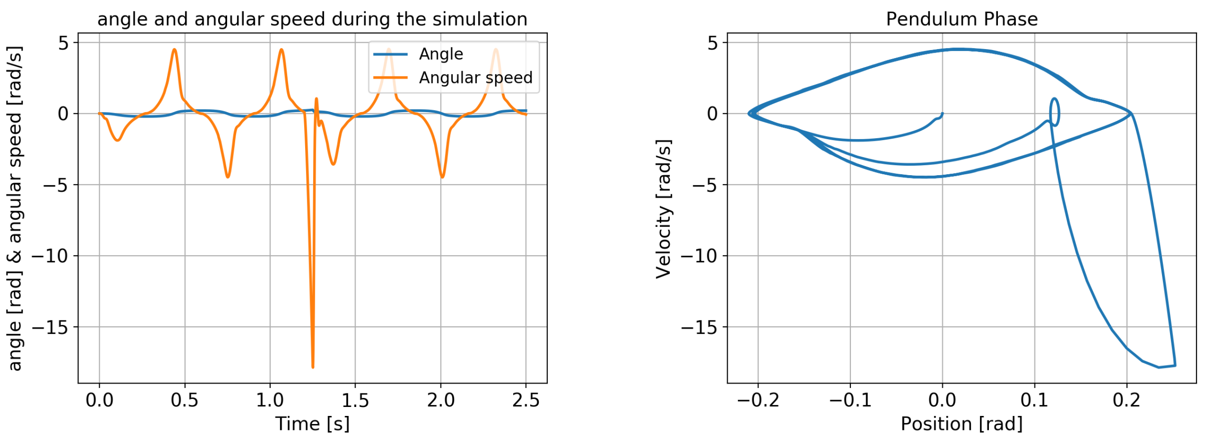


Figure 7: Muscles Activation and Response

In figure 8, we can see the state and phase of our pendulum with a perturbation of $\theta = 1$ between 1.2 and 1.25 seconds. We can see that despite the perturbation, the pendulum regains a stable behavior and exhibits a limit cycle behavior.



(a) State of the pendulum *(b) Phase of the pendulum*

Figure 8: Proof of limit cycle behavior with perturbation

2c. Show the relationship between stimulation frequency and amplitude with the resulting pendulum's behavior.

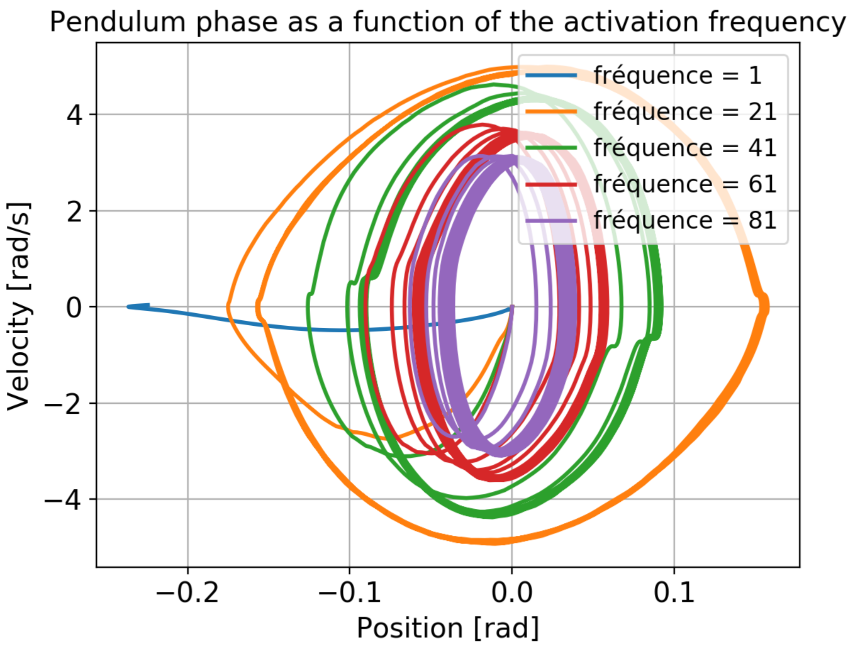


Figure 9: Pendulum Phase for different frequencies

As we can see in Figure 9 that the amplitude of the pendulum shrinks as the stimulation frequency is increased. This makes sense since faster stimulation for the same speed will mean the pendulum will change direction more often and therefore have a smaller amplitude.

Bonnesoeur Maxime
Gautier Maxime
Furrer Stanislas

Exercise 3 : Neural network driven pendulum model with muscles

In this exercise, the goal is to drive the above system 2 with a symmetric four-neuron oscillator network. The network is based on Brown's half-center model with fatigue mechanism. Here we use the leaky-integrate and fire neurons for modelling the network. Figure 10 shows the network structure and the complete system.

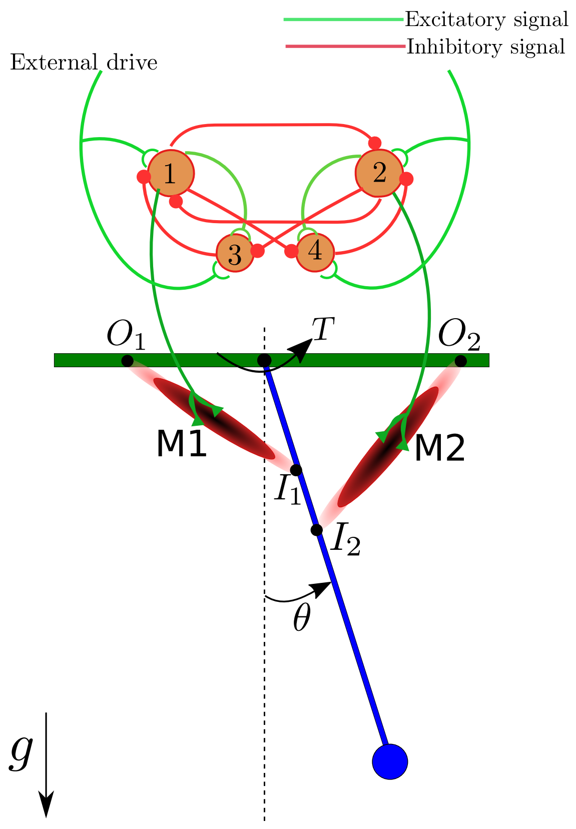


Figure 10: Pendulum with Antagonist Hill Muscles Driven Half Center Neural Network.

Since each leaky-integrate and fire neuron comprises of one first order differential equation, the states to be integrated now increases by four(one state per neuron). The states are,

$$X = \begin{bmatrix} \theta & \dot{\theta} & A_1 & l_{CE1} & A_2 & l_{CE2} & m_1 & m_2 & m_3 & m_4 \end{bmatrix} \tag{6}$$

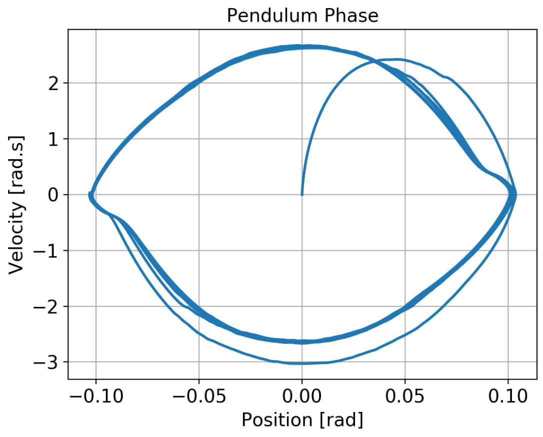
Where,

- m_1 : Membrane potential of neuron 1
- m_2 : Membrane potential of neuron 2
- m_3 : Membrane potential of neuron 3
- m_4 : Membrane potential of neuron 4

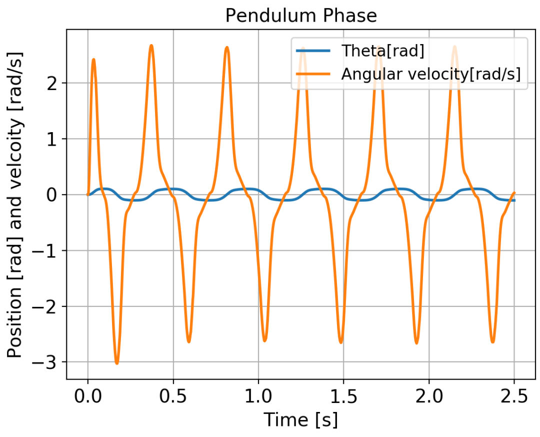
To complete this exercise, additionally you will have to use `neural_system.py` and `exercise3.py`

Bonnesoeur Maxime
Gautier Maxime
Furrer Stanislas

3a. Find a set of weights for the neural network that produces oscillations to drive the pendulum into a limit cycle behavior. Plot the output of the network and the phase plot of the pendulum



(a) Phase of the pendulum



(b) State of the pendulum

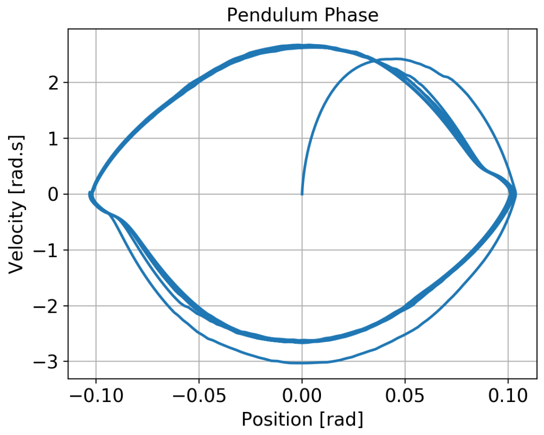
Figure 11: Pendulum State and Phase

To model the oscillating behavior of the neural network, we used the architecture in Figure 10. In fact, the peculiar architecture tells us the type of weight to assign to each of the paths. In the figures 12a and 12b, a symmetric distribution of the weights was used. This means that in our case, the weight of 1 on 2 is the same that the one of two on one as we can see on table 1. we can clearly see in Figure 11 that this network exhibits a limit cycle behavior.

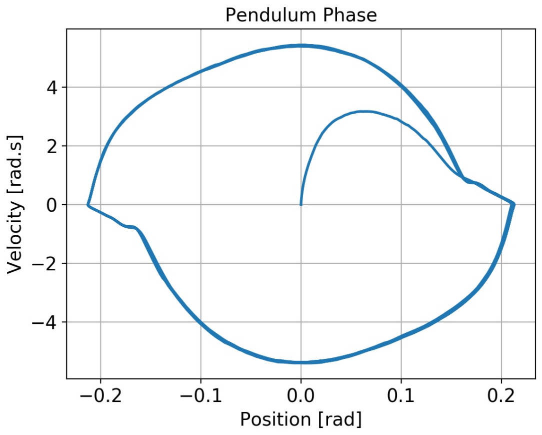
	neuron 1	neuron 2	neuron 3	neuron 4
neuron 1	0	-5	1	-1
neuron 2	-5	0	-1	1
neuron 3	-5	0	0	0
neuron 4	0	-5	0	0

Table 1: Table of the weights of the neurons

3b. As seen in the course, apply an external drive to the individual neurons and explain how the system is affected. Show plots for low [0] and high [1] external drives. To add external drive to the network you can use the method `system_simulation.py::add_external_inputs_to_network`



(a) Pendulum Phase for low external drive



(b) Pendulum Phase for high external drive

Figure 12: Pendulum Phases for varying external drive

As we can see in Figure 12, a high external drive makes the system converge to stable oscillations much quicker, with a greater speed and amplitude. In fact, in figure 12b, a external stimulation of 1 was provided to all the neurons of the system. It will be seen by the system of neurons as an offset of all of their respective values which amplifies the response of the system. This makes perfect sense the external drive is the input of the neural network and works similarly to muscle stimulation in the previous lab : with a higher external drive, the muscles are more stimulated and the resulting produced force is higher.

3c. [Open Question] What are the limitations of the half center model in producing alternating patterns to control the pendulum? What would be the effect of sensory feedback on this model? (No plots required)

The half center model has several advantages: It can be tuned to obtain an oscillation behavior and adding an offset creates an uniform change of both the velocity and position. Nonetheless, this kind of behavior is also a limitation on its own. In fact, not being able to modify the position or velocity independently can be a limitation in some kinds of systems. Moreover, by changing the initialisation values of the neural network, it was found that the system is not always stable and also that receiving external drive on some special neurons tends to destabilize the whole system. In order for this oscillator to be stable, the external drive applied has to be the same for all of the neurons.

As for the effects of sensory feedback, those can be used as an external drive applied to all of the neurons. This way, the system will be able to regulate itself by modifying its velocity and amplitude. With the feedback as input, the input will no longer by fixed and it may allow some more precise control of the system.